

W 1623

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-091706
 (43)Date of publication of application : 29.03.2002

(51)Int.CI. G06F 3/06
 G06F 12/00

(21)Application number : 2001-166124 (71)Applicant : COMPAQ INFORMATION TECHNOLOGIES GROUP LP
 (22)Date of filing : 01.06.2001 (72)Inventor : REUTER JAMES M THIEL DAVID BEAN ROBERT WRENN RICHARD F

(30)Priority

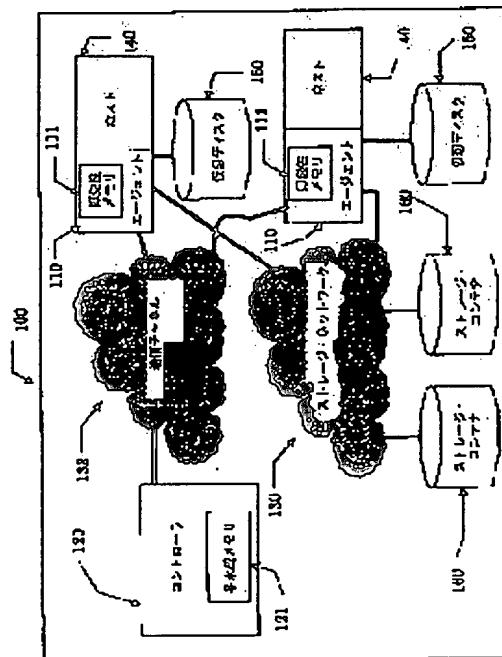
| | | | | | |
|-------------------|-------------|-----------------|------------|--------------------|----|
| Priority number : | 2000 209109 | Priority date : | 02.06.2000 | Priority country : | US |
| | 2000 209326 | | 02.06.2000 | | US |

(54) VIRTUAL STORAGE SYSTEM

(57)Abstract:

PROBLEM TO BE SOLVED: To flexibly perform data access in a distributed environment.

SOLUTION: A host 140 is connected through a network 130 to a storage container 160 and connected through a communication channel 132 to a controller 120. An agent 110 prepares a virtual mapping table by mapping real storage position on the plural storage containers 160 and positions inside a virtual disk 150 and stores it in a volatile memory 111. This table is stored on a semipermanent



memory 121 of the controller as well, and the table in the memory 111 is intermittently replaced with the table in the memory 121. By means of distributed virtualization, the controller can manage a large number of virtual disks to be used for a large number of host systems and a single virtual disk can be shared by a large number of host systems as well.

LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

W16-23

(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開2002-91706

(P2002-91706A)

(43)公開日 平成14年3月29日 (2002.3.29)

(51)Int.Cl.¹G 06 F 3/06
12/00

識別記号

3 0 1
5 4 5

F I

G 06 F 3/06
12/00

マーク (参考)

3 0 1 J 5 B 0 6 5
5 4 5 A 5 B 0 8 2

審査請求 未請求 請求項の数35 OL 外国語出願 (全39頁)

| | |
|-------------|-----------------------------|
| (21)出願番号 | 特願2001-166124(P2001-166124) |
| (22)出願日 | 平成13年6月1日(2001.6.1) |
| (31)優先権主張番号 | 60/209109 |
| (32)優先日 | 平成12年6月2日(2000.6.2) |
| (33)優先権主張国 | 米国(US) |
| (31)優先権主張番号 | 60/209326 |
| (32)優先日 | 平成12年6月2日(2000.6.2) |
| (33)優先権主張国 | 米国(US) |

| | |
|---------|---|
| (71)出願人 | 500039223 コンパック インフォメーション テクノロジーズ グループ リミテッド パーナーシップ Compaq Information Technologies Group, L. P. アメリカ合衆国 テキサス ヒューストン エス エイチ 249 20555 |
| (74)代理人 | 100089705 弁理士 社本 一夫 (外5名) |

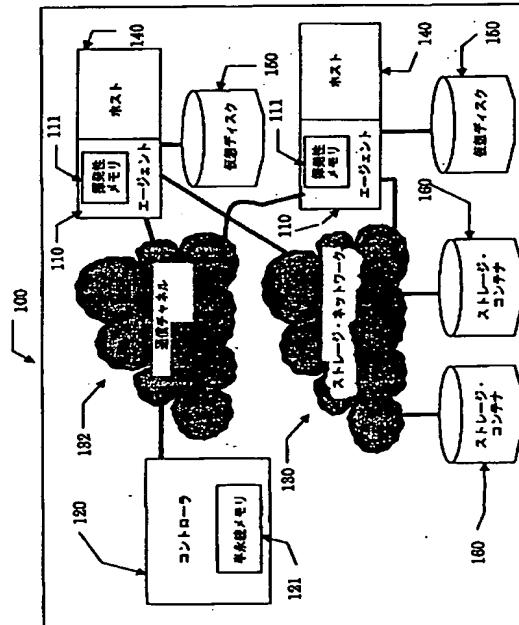
最終頁に続く

(54)【発明の名称】 仮想記憶システム

(57)【要約】

【課題】分散環境においてデータアクセスを柔軟に行う。

【解決手段】ホスト140はネットワーク130を介してストレージ・コンテナ160に、通信チャネル132を介してコントローラ120に接続される。エージェント110は、複数のストレージ・コンテナ160上の実際の記憶箇所と仮想ディスク150内の位置とをマッピングして仮想マッピング・テーブルを作成し、揮発性メモリ111に記憶する。該テーブルはコントローラの半永続メモリ121にも記憶され、メモリ111のテーブルがメモリ121のテーブルと間欠的に置換される。分散仮想化によって、コントローラは、多数のホスト・システムが用いる多数の仮想ディスクを管理することができ、単一の仮想ディスクを多数のホスト・システムで共有することも可能となる。



【特許請求の範囲】

【請求項1】 ネットワークを通じてホストを1以上記憶装置にリンクする仮想記憶システムであって、ホストに接続され、かつ、記憶装置上の記憶箇所に仮想ディスク位置をマッピングするエントリを有するテーブルの第1コピーを記憶する揮発性メモリを有するエージェントと、該エージェントに接続され、かつ、テーブルの第2コピーを記憶する不揮発性メモリを有し、テーブルの第1コピーの内容を、テーブルの第2コピーの内容と間欠的に置換するコントローラとからなり、これにより、入出力(I/O)動作の間、ホストは、エージェント上に記憶されているテーブル内のエントリの1つにアクセスして、記憶装置の記憶箇所の1つを決定することを特徴とする仮想記憶システム。

【請求項2】 請求項1記載のシステムにおいて、テーブルのエントリは更に、無効状態がアクティブ化されているか否かに関する指示を含み、テーブルのエントリが使用可能なマッピング情報を収容していないとき、テーブルの該エントリに対する無効状態がアクティブ化されるよう構成されていることを特徴とするシステム。

【請求項3】 請求項2記載のシステムにおいて、エージェントは、エントリの1つに対する無効状態がアクティブ化されている場合、ホストに該エントリを用いたI/O動作を完了させないように構成していることを特徴とするシステム。

【請求項4】 請求項1記載のシステムにおいて、テーブルのエントリは更に、非書き込状態がアクティブ化されているか否かに関する指示を含み、エントリの1つに含まれる記憶箇所にデータを書き込むことができない場合、該エントリに対する非書き込状態がアクティブ化されるよう構成していることを特徴とするシステム。

【請求項5】 請求項4記載のシステムにおいて、エージェントは、エントリの1つに対する非書き込状態がアクティブ化されている場合、ホストに該エントリにおける記憶箇所にデータを書き込ませないよう構成していることを特徴とするシステム。

【請求項6】 請求項1記載のシステムにおいて、該システムは更に、エージェント及びコントローラに結合された通信チャネルを備えていることを特徴とするシステム。

【請求項7】 請求項6記載のシステムにおいて、通信チャネルは、データ転送プロトコルを用いて通信チャネル上でメッセージをトランスポートするよう構成されていることを特徴とするシステム。

【請求項8】 請求項1記載のシステムにおいて、エントリはオフセットを含むことを特徴とするシステム。

【請求項9】 請求項8記載のシステムにおいて、オフセットは論理ユニット番号識別子を含むことを特徴とするシステム。

【請求項10】 請求項8記載のシステムにおいて、オフセットはブロック識別子を含むことを特徴とするシステム。

【請求項11】 請求項10記載のシステムにおいて、エントリは更に仮想ディスク位置のセグメントを含むことを特徴とするシステム。

【請求項12】 仮想ディスク・セグメントを記憶装置内の記憶箇所にマッピングし、ホストがI/O動作をエージェントに発行し、エージェントが入出力動作に対して記憶箇所を決定するシステムであって、記憶箇所に対応するエントリを有するテーブルと、エントリの状態を示す複数の変数と、エントリに対するオフセットであって、論理ユニット番号識別子及びブロック識別子を含むオフセットと、テーブルを記憶するメモリとからなることを特徴とするシステム。

【請求項13】 請求項12記載のシステムにおいて、メモリが揮発性であることを特徴とするシステム。

【請求項14】 請求項12記載のシステムにおいて、記憶箇所は、記憶装置内のデータ・ブロックであることを特徴とするシステム。

【請求項15】 請求項14記載のシステムにおいて、データ・ブロックは約1MBで構成されていることを特徴とするシステム。

【請求項16】 請求項12記載のシステムにおいて、エージェントがホストに結合されていることを特徴とするシステム。

【請求項17】 請求項12記載のシステムにおいて、複数の変数はブール変数であることを特徴とするシステム。

【請求項18】 請求項12記載のシステムにおいて、エントリの状態は無効状態を含むことを特徴とするシステム。

【請求項19】 請求項18記載のシステムにおいて、複数の変数は無効状態に対する変数を含むことを特徴とするシステム。

【請求項20】 請求項12記載のシステムにおいて、エントリの状態は非書き込状態を含むことを特徴とするシステム。

【請求項21】 請求項20記載のシステムにおいて、複数の変数は非書き込状態に対する変数を含むことを特徴とするシステム。

【請求項22】 請求項12記載のシステムにおいて、エントリの状態はゼロ状態を含むことを特徴とするシステム。

【請求項23】 請求項12記載のシステムにおいて、エントリの状態はエラー状態を含むことを特徴とするシステム。

【請求項24】 ネットワーク内にあるホストに結合された仮想ディスク上で実行される方法であって、

動作中に、仮想ディスク上のブロックを指定するステップと、
ブロックを記憶装置上の記憶箇所にマッピングするテーブルにアクセスするステップと、
記憶装置に対して、仮想ディスク上の動作と相関がある対応する動作を供給する供給ステップと、
対応する動作を完了するステップと、
仮想ディスクに、対応する動作の完了を提示するステップとかなることを特徴とする方法。

【請求項25】 請求項24記載の方法において、供給ステップは、ホストに結合されているエージェントから対応する動作を提供するステップを含むことを特徴とする方法。

【請求項26】 請求項24記載の方法において、該方法は更に、不揮発性メモリに常駐する永続記憶テーブルを用いて、テーブルを更新するステップを含むことを特徴とする方法。

【請求項27】 請求項24記載の方法において、該方法は更に、テーブルの状態を決定するステップを含むことを特徴とする方法。

【請求項28】 請求項24記載の方法において、該方法は更に、テーブルにアクセスすることができない場合、障害メッセージを送るステップを含むことを特徴とする方法。

【請求項29】 請求項24記載の方法において、該方法は更に、不揮発性メモリにテーブルを記憶するステップを含むことを特徴とする方法。

【請求項30】 請求項24記載の方法において、コントローラからテーブルに対する更新を受信するステップを含むことを特徴とする方法。

【請求項31】 ネットワークにおいて仮想ディスク・ブロックを記憶装置上の記憶箇所にマッピングするテーブルを維持する方法であって、
テーブルを記憶するエージェントにおいて、コントローラからコマンドを受信するステップと、
テーブルのエントリ内の状態をアクティビ化するステップと、
テーブルにおいて動作を完了するステップと、
コマンドに応答して、テーブルを更新するステップとかなることを特徴とする方法。

【請求項32】 請求項31記載の方法において、該方法は更に、動作が完了するまで、ブロッキング・フラグをセットするステップを含むことを特徴とする方法。

【請求項33】 請求項31記載の方法において、該方法は更に、テーブルにおけるエントリの1つからマッピング情報を取得するステップを含むことを特徴とする方法。

【請求項34】 ネットワーク内にあるホストに結合された仮想ディスク上で動作を実行するためのコンピュータ読取可能コードが記憶されたコンピュータ使用可能媒

体からなるコンピュータ・プログラム製品であって、該コンピュータ・プログラム製品は、コンピュータ上で、動作中に、仮想ディスク上のブロックを指定するステップと、

05 ブロックを記憶装置上の記憶箇所にマッピングするテーブルにアクセスするステップと、
仮想ディスク上の動作と相関がある対応する動作を記憶装置に供給するステップと、
対応する動作を完了するステップと、

10 仮想ディスクに、対応する動作の完了を提示するステップとを実行するように構成されている、コンピュータ・プログラム製品。

【請求項35】 ネットワークにおいて仮想ディスク・ブロックを記憶装置上の記憶箇所にマッピングするテーブルを維持するためのコンピュータ読取可能コードが記憶されたコンピュータ使用可能媒体からなるコンピュータ・プログラム製品であって、該コンピュータ・プログラム製品は、コンピュータ上で、
テーブルを記憶するエージェントにおいて、コントロー

20 ラからコマンドを受信するステップと、
テーブルのエントリ内において状態をアクティビ化するステップと、
テーブルにおいて動作を完了するステップと、
コマンドに応答して、テーブルを更新するステップとを実行するように構成されているコンピュータ・プログラム製品。

【発明の詳細な説明】

【0001】

【関連出願】 本出願は、2000年6月2日に出願された米国仮出願第60/209,109号及び第60/209,326号からの優先権を主張する。その開示内容は、この言及により全て本願にも含まれるものとする。

【0002】

【発明の属する技術分野】 本発明は、入出力マッピングによる分散型テーブルを用いた、仮想化データ記憶システムに関する。

【0003】

【従来の技術】 単体コンピュータは、通常、図1(A)に概略的に示すように、固定通信チャネル又はバスを通じて、ハード・ディスク、フロッピ・ディスク、テープ、及び光ドライブのようなデータ記憶装置に接続する。このような通信チャネルは高速データ転送を可能にするが、記憶装置に対するアクセスは、単体コンピュータに限定される。

45 【0004】 年々、多数のデバイスを記憶装置に接続し、多数のユーザによるデータ共有化を可能にすることが必要となってきた。その結果、開発者は、図1(B)に概略的に示すように、多数の相互接続され近接配置されたデバイスからなるストレージ・エリア・ネットワーク(SAN)を作成した。SANは、通常、1つ以上の

ネットワーク・アクセス・サーバを備え、SANにおける他のデバイスによるアクセスが可能なデータ記憶装置を含むデバイスの相互接続やネットワークの動作を管理する。これらデバイスは、小型コンピュータ・システム・インターフェース（SCSI）バスを通じて接続すれば、デバイス間に並列通信チャネルを確立することができる。SCSIシステムでは、一意の論理ユニット番号（LUN）を用いてデータ記憶箇所を指定する。各記憶箇所は、別個の記憶装置、又は1つの記憶装置の1区分である。各LUNは更に、小さく容易に管理可能なデータ・サイズのブロックに分割される。LUNゾーニング（zoning）をポート・ゾーニングと組み合わせることによって、SANは、集中、分散データ記憶リソースを有することができる。このSANを介したデータ記憶リソースの共有は、全体のデータ及び記憶管理費用を大幅に削減する。何故なら、記憶装置のコストを多数のデバイス全体で償却することができるからである。また、集中、分散データ・ストレージの使用により、価値あるセキュリティ機能も得られる。これは、SANはデバイスのデータ・アクセス能力を特定のゾーンに制限することができるからである。SAN内では、LUN及び他のネットワーク・デバイス間にファイバ・チャネル接続を用いることによって、高速データ入出力（I/O）動作が達成されるので、統合データ記憶構成を用いる場合の性能コストは、大幅に低減する。実際のところ、SANは、ホスト及びストレージ・コンテナ間の延長及び共有バスとして動作し、とりわけ、記憶管理、スケーラビリティ、柔軟性、可用性、アクセス、移動、及びバックアップの改善がもたらされる。しかしながら、データ・ストレージの集中化によって、データ共有、ストレージ共有、性能最適化、要求に応じた記憶、及びデータ保護を含む新たな問題が生ずる。

【0005】これらの問題のために、最近になって、開発者はSAN階層に仮想化レイヤを追加した。仮想化レイヤとは、実際の記憶装置の物理層又はトポロジには関係なく、使用可能な記憶空間を仮想ディスク又はボリュームに分割するソフトウェア及びハードウェア・コンポーネントのことである。一般に、仮想ボリュームは、物理ディスクのアブストラクション（abstraction）としてサーバのオペレーティング・システムに提示され、仮想ボリュームが物理ディスクであるかのように、サーバによって用いられる。仮想ボリュームは、ストレージ・アレイ上のLUNではない。逆に、仮想ボリュームは、記憶サブシステムとは独立して、作成され、拡張され、削除され、移動され、そして、選択的に提示される。各々は、異なる特性を有し、したがって、使用可能なストレージが拡張するにしたがって、拡張される。SAN仮想化は、オペレーティング・プラットフォームの広い範囲に常駐するアプリケーションに対して、SANリソースの単一プール及び標準的な1組のSAN

サービスを提示する。

【0006】しかしながら、従来のディスク及び記憶サブシステムを用いるSANは、コンピュータ・システム及び記憶装置（ストレージ）間が緊密に結合されているために、かなりのシステム及び記憶管理費用がかかる。これら及びその他の理由のために、既存のSAN技術は、スケーラビリティも限定される。更に、SAN仮想化に残されている主要な問題として、SANの種々のデバイス間におけるストレージ・リソースの分散がある。

05 【0007】したがって、SANにおけるこれら及びその他の要望に対処する、改良されたデータ記憶システムが求められている。提案された記憶システムの1種に、サブシステムを用いて、制御及びアクセス機能をその他の記憶機能から分離することによって、SANの性能を更に向させようとしたものがある。この種のシステムでは、アクセス機能は、SAN上のデータの使用及び操作能力を管理し、制御機能は、デバイス監視、データ保護、及び記憶容量の利用というようなSANの運用に関係する。制御及びアクセス機能を他の記憶機能から分離することによって、仮想化機能がサーバから除外されSANに編入される。従来のサーバ設定実施態様によって設けられるストレージの仮想化に加えて、SAN上の仮想化層によって、仮想ボリュームの作成及び拡張によるデータのコピー、移動及び記憶を含む、重要なデータ移動機能の自動化が可能となる。

【0008】

【発明が解決しようとする課題】制御及びアクセス機能をその他の記憶機能から分離するというこの目的に向かって、現在提案されている仮想化記憶システムは、図2(A)～図2(C)にそれぞれ示すように、ホスト、ストレージ・コントローラ、又はSAN内の特殊な仮想化コンポーネント内というように、集中化した場所に制御及びマッピング機能を統合している。制御及びマッピング機能を集中化することにより、分散マッピングに伴う問題を回避する。しかしながら、1つのコンポーネントに集中させた記憶仮想化方式には、様々なスケーリングの制限が生ずる。これには、多数のコンピュータ・システム、多数の記憶システム、及び適宜の性能を有する大型ストレージ・ネットワークへのスケーリングができないことが含まれる。

【0009】スケーラビリティの改善は、分散仮想化記憶システムによって得ることができる。しかしながら、アレイ・コントローラのような公知の技術を用いて分散仮想化記憶システムを形成し、仮想記憶において用いられるマッピングを分散する試みは、単純なアルゴリズム分散機構を用いており、データ管理の柔軟性、例えば、独立ディスクの冗長アレイ（RAID）が制限される。更に、公知の技術は、ストレージ共有、データ共有、性能最適化、記憶システムの遅延、及びデータ損失の危険性の問題を含む、スケーラブルな仮想記憶システムの課

題に取り組んでいない。

【0010】

【課題を解決するための手段】これらの要望に応じて、本発明は、分散テーブル駆動入出力マッピングを用いたストレージ・エリア・ネットワークにおいて仮想化ストレージを形成するシステム及び方法を提供する。本発明は、コントローラとは別個の多数の並列マッピング・エージェントにおいて、仮想化マッピングを分散する。この構成によって、性能に敏感なマッピング・プロセスを並列化し、性能を最適化するように分散することができ、しかもマッピングの制御は、最適なコスト、管理、及びその他の実施態様の実用性を求めて選択されたコントローラに配置することができる。マッピング・エージェントは、揮発性メモリ内に仮想マッピング・テーブルを記憶することにより、マッピング・エージェントを実現する際のコスト及び複雑性を低減する。コントローラは、マッピング・テーブルの永続的記憶を担うことによって、單一コンポーネント内におけるマッピング・テーブルの永続的記憶に対するコスト及び管理を統合する。また、分散仮想化によって、コントローラは、多数のホスト・システムが用いる多数の仮想ディスクを管理することも可能となり、單一の仮想ディスクを多数のホスト・システムが共有することさえ可能となる。マッピング・エージェントは、他のマッピング・エージェントとは相互作用しないようにすることにより、仮想記憶システムのスケーラビリティ、及び仮想記憶システムのコンポーネント障害に対する耐性を高めることが好ましい。

【0011】

【発明の実施の形態】図3～図6に示すように、本発明は、ホスト140及びストレージ・コンテナ160間の入出力(I/O)動作のために1つ以上の仮想ディスク150を形成するために必要な、1以上の分散マッピング・テーブル200を用いた、仮想ストレージ・エリア・ネットワーク(SAN)を提供する。即ち、テーブル200は、仮想ディスク150内の位置をストレージ・コンテナ160上の実際の記憶箇所と関係付けるマッピングを収容する。テーブル200の具体的な内容については、以下で更に詳しく説明する。

【0012】本発明は、ストレージ・エリア・ネットワーク(SAN)の改良を対象とする。したがって、本発明は、あらゆる既知のストレージ・ネットワーク130に適用可能である。SAN内では、ストレージ・コンテナ160は、既知のものであって、ディスク及びテープ・デバイス、書き可能な光ドライブ等を含むがこれらに限定されない、任意の種類の現行及び未来のプログラム可能デジタル記憶媒体に言及してもよい。同様に、ホスト140は、コンピュータ、プリンタ等、ネットワークに接続ストレージ・コンテナ160からのデータにアクセスする任意のデバイスとすることができます。

【0013】

同様に、ストレージ・ネットワーク130は、小型コンピュータ・システム・インターフェース(SCSI)又はファイバ・チャネルの種々の実施態様のように、現在既知の又は今後開発される、あらゆる通信技術も含むことを想定している。この分散仮想化は、
05 大容量ストレージが使用可能であり、何らかの種類の「ストレージ・ネットワーク」インフラストラクチャを用いて接続される環境において最も有用となる。好適な実施態様の1つでは、ストレージ・ネットワーク130は、切換型ファイバ・チャネル接続ストレージを基本と
10 する。しかしながら、システム100の設計は、未だ発明されていないストレージ・ネットワークを含む他の種類のストレージ・ネットワーク130上における使用も除外するものではない。

【0014】ホスト140は、I/O動作コマンドを仮想ディスク150に発行し、これに応答して、マッピング・エージェント110はテーブル200にアクセスする。エージェント110は一般にはこのようにしてホスト140に関連付けられるが、エージェント110はテーブル200をホスト140のアクセスから隔離させる。好ましくは、ホスト140の各々は別個のエージェント110を有し、各ホストが別個のマッピング・テーブル200を有するようにする。あるいは、システム100は、複数のホストがエージェント110に接続するように構成することも可能である。多数のホスト140が同じエージェント110に接続する場合、ホスト140は、特定のテーブル200に対するアクセスを共有する。

【0015】エージェント110は、典型的にはDRAMである揮発性メモリ111にマッピング・テーブル200を記憶する。その結果、エージェント110の1つがショットダウンすなわち停電した場合、そのエージェント110は自分のテーブル200のコピーを失う。例えば、マッピング・エージェント110がホスト・システム140内に埋め込まれ、その電力を当該ホスト・システムから受ける場合、マッピング・エージェントとして機能するバックプレーン・カードの場合のように、ホスト140は、エージェント110への電力を遮断することによって、マッピング・エージェント110を停止することができる。しかしながら、テーブル200を揮発性メモリに記憶することによって、テーブル200は、エージェント110上で容易にアクセス及び変更が可能となる。更に、マッピング・テーブル200を揮発性メモリに記憶することによって、エージェント110をマッピング・コンポーネントとして実現するコスト及び複雑性を大幅に低減するという別の利点も得られる。全体的に、エージェント110は、性能に敏感なマッピング・プロセスを並列化し、最適な性能が得られるように分散することができる。

【0016】更に、システム100は、コントローラ120を備え、エージェント110とは別個であるが、マ

ッピング・テーブル200を管理しつつ該テーブルをエージェント110に分散する。マッピングの制御は、コスト、管理、及びその他の実施上の実用性の最適化のために、コントローラ120に集中化されている。更に、コントローラ120は、テーブル200を半永続（半永久）メモリ121にも記憶し、シャットダウンの後にもコントローラ120がテーブル200を保持するようとする。半永続メモリ121は、その高記憶容量及び高速かつ頻繁な書き込み能力のため、磁気ディスクが好ましい。あるいは、コントローラ120は、書込可能な光媒体や電子的プログラム可能なメモリのような、他の形態のプログラム可能記憶媒体を用いて、テーブルを記憶してもよい。このように、コントローラ120は、コントローラ120がシャットダウン又は停電しても、テーブル200を記憶し続ける。

【0017】このように、マッピング・テーブル200の永続的記憶は、コントローラ120の役割であり、コスト及び複雑性の双方を統合する。一方、コントローラ120の実際のデザインがこの開示の首題ではなく、この開示では、システム全体の構造、ならびにマッピング・エージェント110及びコントローラ120間のインターフェースが中心である。したがって、デジタル情報記憶の技術分野では既知の任意のコントローラを、本発明を実施する必要性に応じて用いればよい。この枠組みの中で、各エージェント110は、コントローラ120のみと相互作用を行い、他のエージェント110とは行なわないことが好ましい。その結果、システム100は、非常にスケーラブルで、コンポーネントの障害に対する耐性が高くなる。

【0018】以下で説明するように、コントローラ120及びエージェント110の相互作用は、機能及びリターン値で定義される。図3に示す仮想マッピング・システムの一実施形態では、通信は、通信チャネル132のような、ある種のネットワーク・トランスポート上におけるメッセージによって実現される。図4に示すシステム100の別の実施態様では、通信チャネル132は、ストレージ・ネットワーク130自体である。コマンド、障害、及び応答をネットワーク・メッセージに変換するには、適宜の技法であれば任意のものを用いることができる。通信チャネル130は、TCP/IPのような既知のデータ転送プロトコルであれば、任意の形式でも使用可能である。コントローラ120の機能及びアクティビティ間の個々の相互作用については、以下で更に詳しく説明する。

【0019】図5及び図6は、テーブル200の内容を概略的に示している。前述のように、テーブル200は、1つ以上の仮想ディスク・セグメント220とストレージ・コンテナ160上の記憶箇所230との間のマッピングを含むエントリ210（行）を含む。記憶箇所230は、個々のストレージ・コンテナ160、及び仮

想ディスク・インデックス220に対応するストレージ・コンテナ160の部分を特定する。記憶箇所230の形態は、用いるストレージ・ネットワークに対して適切でなければならない。SCSIネットワークでは、記憶箇所230の各々は、LUN識別子233と、オフセットとも呼ぶブロック識別子235とを含む。マッピング・テーブル・エントリ210内の他のフィールド全ては、単純な整数又は二進的な状態値である。

【0020】図5に示すように、マッピング・テーブル200は、仮想ディスク220の各「ディスク・ブロック」毎に1つのエントリ210を有することができる。これは構築可能ではあるが、マッピング・テーブルが巨大となり、非常に断片化したマッピングとなし、望ましくない能力低下を招く。別の実施形態では、各マッピング・テーブル・エントリ210は、物理ストレージ・コンテナ160の1つにおける隣接ブロックにマッピングする、可変サイズの隣接仮想ディスク・ブロック・グループを表す。このテーブル200の構成は、マッピング柔軟性の向上及び高密度のマッピング構造をもたらすが、可変サイズ範囲を管理する上でのアルゴリズムの複雑性、及びマッピング・エントリ検索の際の高コスト化を招く。

【0021】一方、好適な実施形態のテーブル200は、マッピング・テーブル・エントリ210を用いており、各々、図6に示すように、1つのストレージ・コンテナ160にマッピングする仮想ディスク150上に固定サイズ数の隣接ブロック（「セグメント」）を有する。この実施形態では、エントリ210の各々は、仮想ディスク・ブロックの代わりに、仮想ディスク・セグメント220を収容する。ブロック識別子235は、同様に、実際の記憶ブロックの対応するセグメントを識別する。図6は、マッピング・テーブル・エントリ220及びブロック識別子235に対する値の全範囲を特定するテーブル200を示すが、テーブル200は同様に先頭又は末尾ブロックのみを特定することも可能である。この場合、実際の記憶セグメント及び仮想記憶セグメントのサイズは、他の方法で定義する。図6のテーブル200に対する構成は、恐らく可変サイズ範囲マッピング程度が高くないが、この構成は、最も単純で最も高い性能のマッピング・アクセス及び空間管理をもたらす。テーブル200の具体的な詳細には無関係に、該テーブル200は、仮想ディスク・セグメント220を、I/O動作に関与する各物理記憶ブロックにマッピングしなければならない。

【0022】好適な実施形態では、システム100は、多数のテーブル200を有し、その各々は、仮想ディスク150及びストレージ・コンテナ160間で異なるマッピングを行なう。このように、異なるホスト140は、同じストレージ・コンテナ160に対して異なるアクセスを行なうことができる。テーブル200が記憶箇

所230の1つを含まない場合、このテーブルを用いるホスト140（即ち、このテーブルを記憶するエージェント110に接続するホスト140）は、その記憶箇所に記憶されている情報にアクセスすることはできない。実際のところ、ホスト140は、この記憶箇所230が存在することを認識することができない。

【0023】動作中、ホスト140はI/O動作（例えば、リード又はライト）を、仮想ディスク150上のいずれかのブロック又は複数のブロックに発生させる。各仮想メモリ・ブロックは、マッピング・テーブル200において、個々のエントリとして又は仮想ディスク・セグメント220の一部として表される。I/O動作に含まれる各ブロックは、ストレージ・コンテナ160上の適切な記憶箇所にマッピングされる。マッピング・エージェント110は、ストレージ・コンテナ160に対して、対応するI/O動作を発生する。次に、I/O動作の結果が、収集されかつ仮想ディスク150上において完了した動作として提示される。

【0024】記憶箇所を指定するマッピング情報に加えて、各マッピング・テーブルのエントリ210は、いくつかの状態も含む。これらの状態は、仮想ディスク・セグメントの現在ステータスに関する情報を与える、ブル変数である。これらの状態は、エージェント110内に記憶されているテーブル200を遠方にロードしたり、コントローラ120から操作することを可能にするので、重要である。これらの状態及びインターフェースは、マッピング・テーブルを分散し、マッピング・テーブルのエントリを揮発性にすることを可能にする。本明細書では、最初に、状態について説明し、次いで状態に対する関数の一部について説明する。テーブル200は、何らかのI/O動作が仮想ディスク・セグメント220上で生じたか否かを示す少なくとも1つの無効状態240、及びその対応する物理的位置（位置）230を含む。無効状態は、第1のI/O動作の間にアクティブ化され、この第1のI/O動作の完了まで別のI/O動作を禁止することができる。好適な実施形態では、テーブル200は更に、非書き込み（Nw）状態250を含む。これは、対応する物理的位置230に収容されているデータを現在変更してもよいか否かについて示す。Nw状態250によって、記憶システムの性能向上が可能となる。何故なら、これは他のI/O動作中にデータを読み取ることを可能にするからである。無効状態240及びNw状態250は、マッピング・テーブルのエントリの動的ローディング、動的マッピング変更、マッピング・テーブルのエントリの揮発性、及び同様の仮想ディスク150間のデータ共有の間、機能する。

【0025】無効状態240は、アクティブ化された場合は通常、マッピング・テーブルのエントリ210が使用可能なマッピング情報を収容しておらず、I/O動作に対応できることを示す。このテーブルのエントリ2

10によってI/O動作を実行しようとすると、そのいずれの場合でも、マッピング・エージェント110は障害メッセージをコントローラ210に送る。エージェント110は、コントローラ120が障害応答を返送するまで、I/O動作を開始しない。一つの実施形態では、システム100は、テーブル200が新たに作成されたとき、最初にテーブル200内のエントリ210全てに對して無効状態240をアクティブ化する。このように、テーブル200は、以前に記憶されていたテーブルからの、メモリ内のあらゆる残留エントリを無視し、現在のエントリがアクティブであり信頼性があることを保証する。同様に、エントリ210が「忘れられ」、マッピング・エージェント110の揮発性メモリによって失われたときに、無効状態240をアクティブ化することもできる。無効状態240がエントリ210においてアクティブ化された場合、エントリ210内にある他の値及び状態の全ては、有効な情報を収容していないと想定され、無視される。

【0026】マッピング・エージェント110内に位置するテーブル200が揮発性であるので、マッピング・エージェント110に障害が発生したり、再起動する場合は常に、エントリ210の全てがアクティブな無効状態240を有することになる。コントローラ120及びマッピング・エージェント110間の通信喪失が継続すると、マッピング・テーブルのエントリ全てをアクティブな無効状態240に戻すことによって、あるいはコントローラ120においてI/O動作を再開するように指示されるまでI/O動作を保留にしておく追加の機構を付加することによって、I/O動作が停止する。この構成により、コントローラ120は、異常なマッピング・エージェント110、又は到達不可能なマッピング・エージェント110が既知の状態に置かれたことを知ることによって、他のマッピング・エージェント110を調整し続けることができ、残存するマッピング・エージェント110に対するデータ・アクセスの高い可用性を、コントローラ120に提供する。

【0027】先に提示したように、Nw状態250は、アクティブ化されると、エントリ210によって示される仮想ディスク・セグメント220に対するあらゆる書き込動作によって、コントローラ120はエージェント110に障害メッセージを送ることを示す。エージェント110は、コントローラ120が障害応答を返送してNw状態250を非アクティブ化するまで、又はシステム100がその他の方法で、Nw状態250がアクティブであることにも拘らず、セグメントに書き込むための何らかの処置を講ずるまで、ホスト140に記憶箇所230に書き込むことを許さない。無効状態240とは異なり、アクティブ化されたNw状態250は、動作が障害を発生するのを防止する。一方、エージェント110は、通常、ホスト140が処理を先に進めて記憶箇所2

30のデータにアクセスすることができるようになる。したがって、Nw状態のみがアクティブ化されている場合、テーブル・エントリ210は、使用可能な記憶箇所230を含んでいなければならない。

【0028】別の実施形態では、テーブル200は、更に、ゼロ(Z)状態260を含む。アクティブの場合、Z状態260は、エントリ210が示す仮想ディスク・セグメント220が全て0のバイトを含んでいることを示す。この機能によって、仮想ディスク150が形成され、非仮想ストレージの割り当てや調整を全く必要とせずに、初期化したように見せることができる。エントリ210がアクティブなZ状態260を収容している場合、エージェント110は、その記憶アドレス230を無視する。ホスト140が記憶アドレス230に記憶されている情報を読み出そうとすると、エージェント110は、記憶アドレス230の実際の内容には関係なく、0で満たされたブロックのみを返送する。一方、Z状態260がアクティブなときに記憶アドレス230にデータを書き込もうとしたときはいつでも、エージェント110は障害メッセージをコントローラ120に送る。エージェント110は、コントローラ120が障害応答を返送してZ状態260を非アクティブ化するまで、又はシステム100がその他の方法で、Z状態260がアクティブであることにも拘らず、セグメントに書き込む何らかの処置を構ずるまで、ホスト140に記憶箇所230に書き込むことを許さない。

【0029】別の構成では、マッピング・テーブル200は、更に、エラー(E)状態270も含む。E状態270は、アクティブの場合にエラー状態の存在を示し、以前の状態を全く破壊することなくエージェントにエラーを返送するように指示するために必要な情報を与える。E状態270は、既存の障害がわかった場合に用いられ、このような障害により、I/Oアクセスのあらゆる試行は失敗に終わる。しかしながら、E状態270は、マッピング障害からエラー状態を発生する手段としても使用可能であることを注記しておく。エントリ210がアクティブなE状態270を含む場合、エージェント110は記憶アドレス230を無視する。ホスト140が記憶アドレス230に対して読み出又は書き込を行なうとした場合、エージェント110は、エラーをホスト140に返送する。エージェント110及びコントローラ120の相互作用について、これより詳細に説明する。相互作用の一分類、障害/応答動作では、エージェント110はメッセージをコントローラ120に送り、テーブル200に対するI/O動作中に障害が発生したことを見出す。典型的に、障害は、エージェントによるI/O動作の実行を妨げるアクティブ化された状態の結果として発生する(前述通り)。エージェント110は、障害メッセージをコントローラ120に送る。次いで、コントローラは、適切な処置を決定し、それに応じてエージェント110に指令する。

【0030】障害/応答動作の一形態、マッピング障害では、マッピング・エージェント110は、マッピング・テーブルのエントリ210がアクティブ状態を有して要求されたI/O動作の完了を妨げるので、ホスト140によって要求されたI/O動作を完了できないことを、コントローラ120に警告する。例えば、マッピング・エージェント110は、アクティブな無効フラグ240を有するテーブル・エントリ210に対するあらゆるI/O動作の要求に応答して、又はアクティブな対応するNwフラグ250を有する記憶アドレス230への書き込み試行に応答して、コントローラ120への障害メッセージを生成する。エージェント110からのマッピング障害メッセージは、通常、要求されたI/O動作、関与する仮想ディスク・セグメント220、及びI/O動作を妨げるテーブルの状態を特定する。障害が発生すると、エージェントはI/O動作を実行しようとしている。一方、コントローラ120は、障害メッセージを用いて、異常I/O動作(例えば、マッピング・エントリのローディング、マッピング・エントリの変更、他の何らかの動作が完了するまでの遅延)に応答する。コントローラ120の応答は、マッピング・エージェント110に、どのようにして障害の原因を克服するかについて通知する。

【0031】コントローラ120は、通常、問題を解決するか、又は要求元のホストにエラー・メッセージを送るように、エージェント110に命令する。問題を解決する場合、コントローラ120は、テーブルの置換すべきエントリ210を送る。次いで、エージェント110は新たなエントリ210をテーブルに(以前の異常エントリの代わりに)挿入し、次いでI/O動作を再度試行する。コントローラ120が問題を解決できない場合、エラー・メッセージをホストに発生するように、エージェント110に命令する。エージェント110にエラー・メッセージを発生させるために、コントローラは、エージェントに、障害の原因となったテーブルのエントリ210に対するエラー状態260をアクティブ化するように指令する。すると、前述のように、テーブルのエントリ210の他の内容には無関係に、エージェント110は、エラー・メッセージをホスト140に対して発生する。

【0032】コントローラ120によって発せられるエージェント110に対するコマンドは、第2のカテゴリの相互作用、コマンド/応答動作からなる。これらコントローラ120によって発せられるコマンドは、新たなマッピング・テーブル200の作成を含み、全てのエントリは、アクティブな無効フラグを有するようにセットされるか、又は既存のテーブル200を削除する。コントローラ120は、エージェント110から、エントリ210の1つの内容、又はこのエントリ210における

状態の1つのステータスを取得することができる。更に、コントローラ120は、エントリ210の1つに対する内容の全て、又はエントリ210に対する状態の1つのステータスをセットするように、エージェント110に指令する。なお、無効状態240、エラー状態260、及びゼロ状態270がアクティブとなったなら、前述のように、これらの状態の初期アクティビ化が記憶アドレス230を無効にするので、コントローラ120はその状態を非アクティビ化する。これらの状態を非アクティビ化するために、コントローラ120は、既存のエントリ210を全く新しいエントリと置換するように、エージェント110に指令しなければならない。これらのコマンドの各々に対して、エージェントは、指令されたタスクを完了した後に、コントローラ120に応答を返送する。

【0033】コントローラがテーブル200に情報をセットするか又はテーブル200から情報を取得するようにエージェントに命令したとき、システムは、コントローラ120に単一コマンドにおいて多数の隣接するマッピング・テーブル・エントリ210を指定させることができ最適である。この構成によって、エージェント110及びコントローラ120が一層効率的に、しかもより少ない命令で、相互作用することが可能となる。

【0034】コントローラ120がエントリ・テーブル210内の値及び状態の1つ又は全てをセットするようにエージェント110に指令するとき、コントローラ120は、ブロッキング・フラグを最適に含むようにエージェント110に命令する。ブロッキング・フラグは、コマンドの前に開始されたあらゆるI/O動作の完了後までコントローラ120のコマンドに応答するのを延期するように、エージェント110に指示する。ブロッキング・フラグは、それが常駐するコマンドに対してのみ適用される。通常、他の同時のコマンド及び後続のコマンドは、1つのコマンドのブロッキング・フラグの影響を受けない。即ち、エージェント110は、コマンドで指令されるように、直ちにテーブル200を変更するが、以前から存在するI/O動作全てが完了するまで、この変更をコントローラ120に通知しない。このように、エージェント110は、コマンド内に指定されたテーブルに対する変更を反映しない以前のテーブル200を用いて、全てのI/O動作の完了を、コントローラ120に連絡する。

【0035】動作の大部分において、マッピング・エージェント110は障害なく動作する。無障害の場合、即ち、マッピング・テーブル・エントリ210が有効であり、要求されたI/O動作を妨げるアクティブ状態が全くない場合、仮想ディスクI/Oは、マッピング・エージェント110を通じて完全に動作する。したがって、全てのI/O動作は、マッピング・テーブル200を介して、物理ストレージ・コンテナ160まで進み、コントローラ120の関与は全くない。その結果、コントローラ120は、種々の管理動作を行なわなければならないときにのみ、それ自体をI/Oストリームに挿入し、通常は無障害の場合には関与せず、システム100は高性能及び高スケーラビリティを有することができる。

【0036】本発明の好適な実施形態に関するこれまでの記載は、例示及び説明のために提示したのである。本発明がこれで全てであるという訳ではなく、即ち開示した正確な形態に本発明を限定することを意図する訳ではない。前述の教示にしたがって多くの変更や変形が可能である。本発明の範囲は、この詳細な説明によって限定されるのではなく、ここに添付した特許請求の範囲によって限定されることとする。前述の明細書、例及びデータは、本発明の構成体の製造及び使用について完全な説明を提供するものである。本発明の多くの実施形態が本発明の精神及び範囲から逸脱することなく行なうことができるので、本発明は以下に添付する特許請求の範囲に帰すこととする。

【図面の簡単な説明】

【図1】ホストを記憶装置に接続する既知のシステムを示す図である。

【図2】既知の公知の仮想化ストレージ・エリア・ネットワークを示す図である。

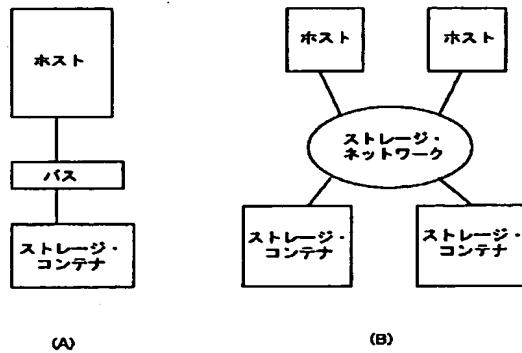
【図3】本発明の一実施形態による分散仮想ストレージ・エリア・ネットワークの概略図である。

【図4】本発明の別の実施形態による分散仮想ストレージ・エリア・ネットワークの概略図である。

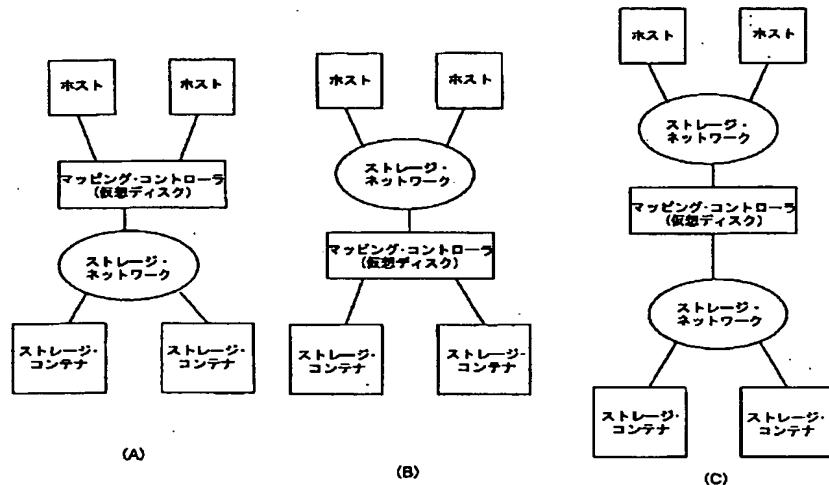
【図5】本発明の実施形態に係る図3の分散仮想ストレージ・エリア・ネットワークにおいて使用するためのマッピング・テーブルの概略図である。

【図6】本発明の実施形態に係る図4の分散仮想ストレージ・エリア・ネットワークにおいて使用するためのマッピング・テーブルの概略図である。

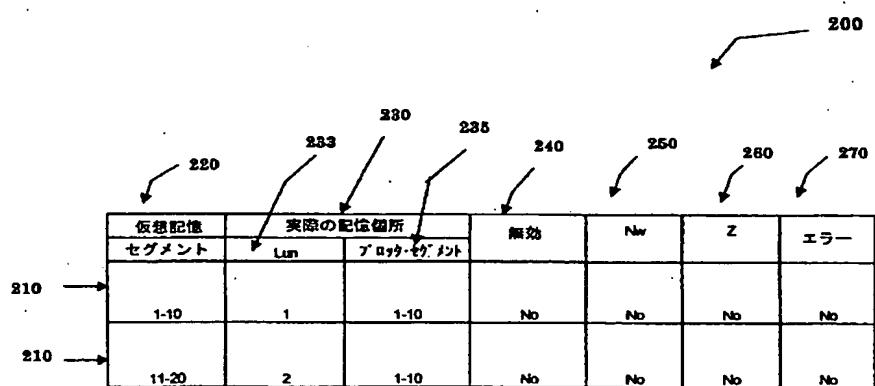
【図1】



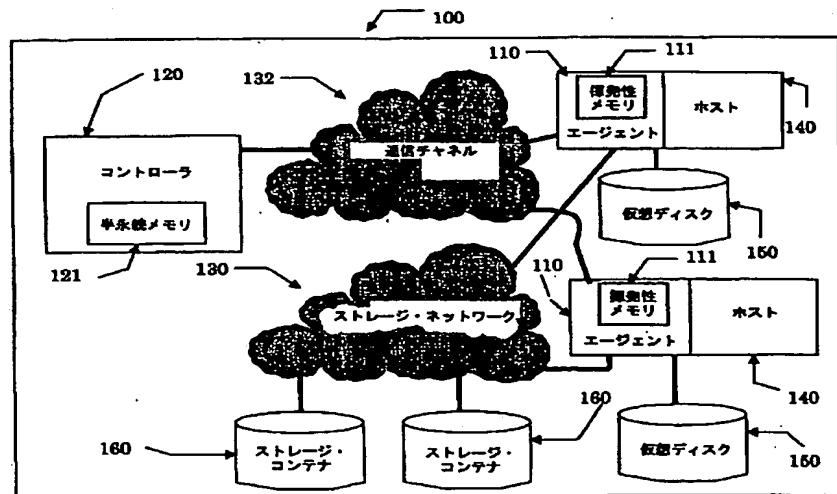
【図2】



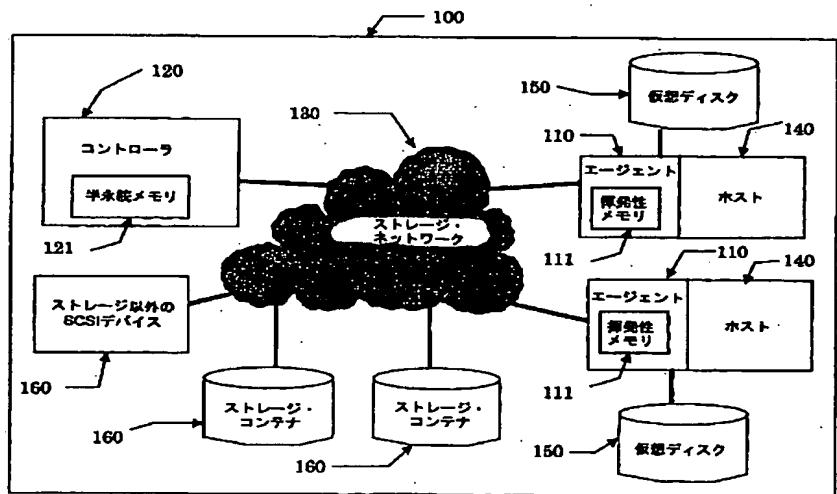
【図6】



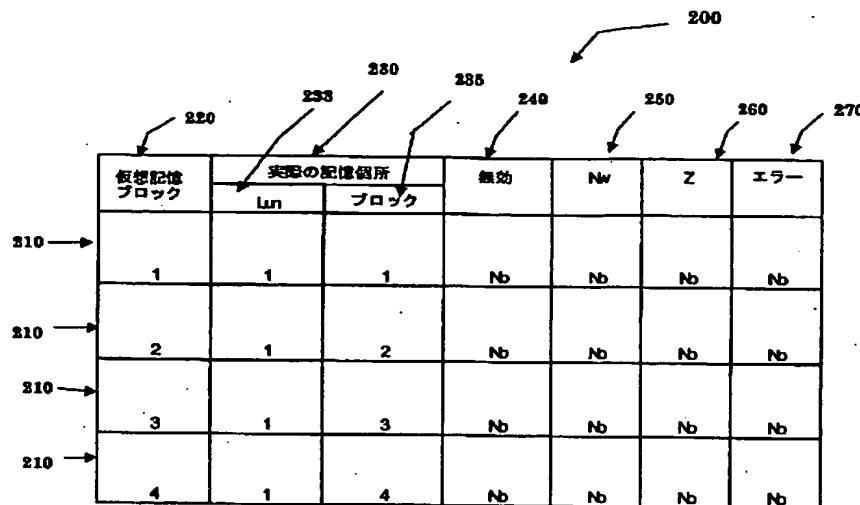
【図3】



【図4】



【図5】



フロントページの続き

(71)出願人 500039223 25 (72)発明者 ロバート・ピーン
 20555 S. H. 249, Houston
 n, Texas 77070, U. S. A.

(72)発明者 ジェームズ・エム・ロイター (72)発明者 リチャード・エフ・レン
 アメリカ合衆国コロラド州80919, コロラ
 ド・スプリングス, ダンテ・ウェイ 7722 30
 アメリカ合衆国コロラド州80919, コロラ
 ド・スプリングス, オーク・ヒルズ・ドラ
 イブ 2360

(72)発明者 デーヴィッド・シール
 アメリカ合衆国コロラド州80919-4524,
 コロラド・スプリングス, ニードルコー
 ン・レイン 2935 Fターム(参考) 5B065 BA06 CC01 ZA08
 5B082 HA01 HA05

【外国語明細書】

1. Title of Invention

Virtual Storage System

2. Claims

1. A virtual storage system for linking a host to one or more storage devices over a network, the system comprising:
an agent connected to the host, the agent having volatile memory for storing a first copy of a table, the table having entries to map virtual disk positions to locations on the storage devices; and
a controller coupled to the agent, the controller having non-volatile memory for storing a second copy of the table, the controller intermittently causing contents of the first copy of the table to be replaced by contents of the second copy of the table,
whereby during an input/output (I/O) operation, the host accesses one of the entries in the table stored on the agent to determine one of the storage device locations.
2. The system of claim 1, wherein the table entries further include an indication of whether an invalid state is activated such that the invalid state for a table entry becomes activated when that table entry contains no useable mapping information.
3. The system of claim 2, wherein the agent does not allow the host to complete the I/O operations with one of the entries if the invalid state for that entry is activated.
4. The system of claim 1, wherein the table entries further include an indication of whether a no-write state is activated such that the no-write state for one of the entries becomes activated when data cannot be written to the storage location contained in that entry.
5. The system of claim 4, wherein the agent does not allow the host to write data to the storage location in one of the entries if the no-write state for that entry is activated.

6. The system of claim 1, further comprising a communication channel to couple the agent and the controller.

7. The system of claim 6, wherein the communication channel employs a data transfer protocol to transport messages on the communication channel.

8. The system of claim 1, wherein the entries include an offset.

9. The system of claim 8, wherein the offset includes logic unit number identifier.

10. The system of claim 8, wherein the offset includes a block identifier.

11. The system of claim 10, wherein the entries further includes a segment of virtual disk positions.

12. A system for mapping a virtual disk segment to a storage location within a storage device, such that a host issues a I/O operation to an agent and the agent determines said storage location for input/output operations, said system comprising:

- a table having an entry corresponding to said storage location;
- a plurality of variables indicating states of the entry;
- an offset for the entry, wherein the offset includes a logic unit number identifier and a block identifier; and
- a memory to store the table.

13. The system of claim 12, wherein the memory is volatile.

14. The system of claim 12, wherein said storage location comprises a block of data within the storage device.

15. The system of claim 14, wherein the block of data is about 1 MB.

16. The system of claim 12, wherein the agent is coupled to the host.
17. The system of claim 12, wherein the plurality of variables comprise Boolean variable.
18. The system of claim 12, wherein the states include an invalid state.
19. The system of claim 18, wherein the plurality of variables includes a variable for the invalid state.
20. The system of claim 12, wherein the states include a no-write state.
21. The system of claim 20, wherein the plurality of variables includes a variable for the no-write state.
22. The system of claim 12, wherein the states include a zero state.
23. The system of claim 12, wherein the states include an error state.
24. A method for performing an operation on a virtual disk coupled to a host within a network, comprising:
 - specifying a block on the virtual disk within the operation;
 - accessing a table mapping the block to a storage location on a storage device;
 - issuing a corresponding operation to the storage device, wherein the corresponding operation correlates to the operation on the virtual disk;
 - completing the corresponding operation; and
 - presenting the completed corresponding operation to the virtual disk.

25. The method of claim 24, wherein the issuing step includes issuing the corresponding operation from an agent coupled to the host.

26. The method of claim 24, further comprising updating the table with a persistently-stored table residing in a non-volatile memory.

27. The method of claim 24, further comprising determining states of the table.

28. The method of claim 24, further comprising sending a fault message when the table is unable to be accessed.

29. The method of claim 24, further comprising storing the table in a volatile memory.

30. The method of claim 24, receiving updates for the table from a controller.

31. A method for maintaining a table for mapping virtual disk blocks to storage locations on storage devices within a network, comprising:

receiving a command from a controller at an agent storing the table;
activating states within entries of the table;
completing operations at the table; and
updating the table in response to the command.

32. The method of claim 31, further comprising setting a blocking flag until operations are completed.

33. The method of claim 31, further comprising obtaining mapping information from one of the entries in the table.

34. A computer program product comprising a computer useable medium having computer readable code embodied therein for performing an operation on a virtual disk coupled to a host within a network, the

computer program product adapted when run on a computer to effect steps including:

specifying a block on the virtual disk within the operation;
accessing a table mapping the block to a storage location on a storage device;
issuing a corresponding operation to the storage device, wherein the corresponding operation correlates to the operation on the virtual disk;
completing the corresponding operation; and
presenting the completed corresponding operation to the virtual disk.

35. A computer program product comprising a computer useable medium having computer readable code embodied therein for maintaining a table for mapping virtual disk blocks to storage locations on storage devices within a network, the computer program product adapted when run on a computer to effect steps including:

receiving a command from a controller at an agent storing the table;
activating states within entries of the table;
completing operations at the table; and
updating the table in response to the command.

3. Detailed Description of Invention

Related Applications

This application claims priority from U. S. Provisional Application Nos. 60/209,109 and 60/209,326, filed on June 2, 2000, the disclosures of which are hereby incorporated by reference in full.

Field Of The Invention

The present invention is a virtualized data storage system using distributed tables with input/output mappings.

Background Of The Invention

A stand alone computer generally connects to data storage devices, such as hard disk, floppy disk, tape, and optical drives, via a fixed communication channel or bus, as schematically illustrated in FIG. 1A. While the communication channel allows high-speed data transfers, access to the storage device is limited to the stand-alone computer.

Over time, it has become necessary for multiple devices to connect to a storage device so that multiple users may share data. As a result, developers created a storage area network (SAN) consisting of multiple, interconnected, proximately located devices, as schematically illustrated in FIG. 1B. The SAN typically includes one or more network access servers that administer the interaction of the devices and the operation of the network, including data storage devices that are accessible by the other devices in the SAN. The devices may be connected through Small Computer Systems Interface (SCSI) buses to establish parallel communication channels between the devices. In SCSI systems, a unique Logical Unit Number (LUN) is used to designate data storage locations, where each location is a separate storage device or partition of a storage device. Each LUN is further divided into blocks of small, easily manageable data sizes. By combining LUN zoning with port zoning to

implement storage sharing, the SAN can have centralized, distributed data storage resources. This sharing of data storage resources across the SAN substantially reduces overall data and storage management expenses, because the cost of the storage devices may be amortized across multiple devices. The use of centralized, distributed data storage also provides valuable security features because the SAN may limit the ability of a device to access data in a particular zone. The performance costs of using consolidated data storage configurations within the SAN are substantially reduced through the use of Fibre Channel connections between the LUNs and the other network devices to achieve high-speed data input and output (I/O) operations. The SAN operates, in effect, as an extended and shared storage bus between the host and the storage containers to offer, among other things, improved storage management, scalability, flexibility, availability, access, movement, and backup. The centralization of data storage, however, presents new problems, including issues of data sharing, storage sharing, performance optimization, storage on demand, and data protection.

Because of these issues, developers have recently added a virtualization layer to the SAN hierarchy. The virtualization layer refers to software and hardware components that divide the available storage spaces into virtual disks or volumes without regard to the physical layer or topology of the actual storage devices. Typically, virtual volumes are presented to the server operating system as an abstraction of the physical disk and are used by the server as if virtual volumes were physical disks. The virtual volumes are not LUNs on a storage array. Instead, the virtual volumes may be created, expanded, deleted, moved, and selectively presented, independent of the storage subsystem. Each has different characteristics, and therefore expanded as the available storage expands. The SAN virtualization presents a single pool of SAN resources and a standard set of SAN services to applications residing on a broad range of operating platforms.

However, SANs using conventional disks and storage subsystems incur substantial system and storage management expenses due to the tight coupling between the computer system and the storage. Because of these and other reasons, the existing SAN technologies also have limited scalability. Furthermore, a key remaining issue for SAN virtualization is the distribution of storage resources among the various devices of the SAN.

Accordingly, there exists a need for an improved data storage system that addresses these and other needs in the SAN. One proposed class of storage system uses a subsystem to further improve the performance of the SAN by separating control and access functions from other storage functions. In such a class, access functions govern the ability to use and manipulate the data on the SAN, and control functions relate to the administration of the SAN such as device monitoring, data protection, and storage capacity utilization. Separating control and access functions from other storage functions pulls the virtualization function out of the server and onto the SAN. In addition to the virtualization of the storage provided by traditional, server bound implementations, the virtualization layer on the SAN enables the automation of important data movement functions, including the copying, movement, and storage of data through the creation and expansion of virtual volumes.

Toward this purpose of separating control and access functions from other storage functions, currently proposed virtualized storage systems consolidate control and mapping functions in a centralized location such as in the host, in a storage controller, or in a special virtualization component in the SAN, as illustrated in FIGS. 2A-2C respectively. Centralizing the control and mapping functions avoids problems associated with distributed mapping. However, storage virtualization schemes that are centralized in one component suffer from various scaling limitations, including the inability of scaling to multiple computer systems, multiple storage systems, and large storage networks with adequate performance.

Improved scalability may be achieved through a distributed virtualized storage system. However, attempts to form distributed virtualized storage systems through the use of known technologies, such as array controllers, for distributing the mapping used in the virtual storage use simple algorithmic distribution mechanisms that limit data management flexibility, e.g. Redundant Array of Independent Disk (RAID). Furthermore, the known technologies do not address the needs of a scaleable virtual storage system, including issues of storage sharing, data sharing, performance optimization, storage system delays, and data loss risks.

SUMMARY OF THE INVENTION

In response to these needs, the present invention provides a system and method for creating virtualized storage in a storage area network using distributed table-driven input/output mapping. The present invention distributes the virtualization mapping in multiple parallel mapping agents that are separate from a controller. This configuration allows the performance-sensitive mapping process to be parallelized and distributed optimally for performance, while the control of the mapping can be located in the controller chosen for optimal cost, management, and other implementation practicalities. The mapping agents store the virtual mapping tables in volatile memory, substantially reducing the cost and complexity of implementing the mapping agents. The controller is responsible for persistent storage of mapping tables, thereby consolidating the costs and management for persistent mapping table storage in a single component. Distributed virtualization also allows the controller to manage multiple virtual disks used by multiple host systems, and even allows a single virtual disk to be shared by multiple host systems. The mapping agents preferably do not interact with other mapping agents, thereby improving the scalability of the virtual storage system and the virtual storage system's tolerance of component failures.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

As illustrated in FIGS. 3A-3B and 4A-4B, the present invention provides a virtualized storage area network (SAN) system 100 using one or more distributed mapping tables 200, as needed to form one or more virtual disks 150 for input/output (I/O) operations between the hosts 140 and storage containers 160. In particular, the table 200 contains a mapping that relates a position in the virtual disk 150 with the actual location on the storage containers 160. The specific contents of the table 200 are described in greater detail below.

The present invention covers an improved storage area network (SAN). The invention can therefore be applied to any known storage network 130. Within the SAN, it should be appreciated that the storage containers 160 are known and may refer to any type of present or future known programmable digital storage medium, including but not limited to disk and tape drives, writeable optical drives, etc. Similarly, the hosts 140

may be any devices, such as a computer, printer, etc. that connect to a network to access data from a storage container 160.

Likewise, the storage network 130 is also intended to include any communication technology, either currently known or to be developed in the future, such as the various implementations of Small Computer Systems Interface (SCSI) or Fibre Channel. This distributed virtualization is most useful in environments where a large amount of storage is available and connected using some sort of "storage network" infrastructure. In one preferred implementation, the storage network 130 is based on Switched Fibre-Channel connected storage. However, nothing in the design of the system 100 precludes its use on other types of storage networks 130, including storage networks that are not yet invented.

The hosts 140 issues I/O operation commands to the virtual disks 150, and in response, mapping agents 110 access the table 200. Although the agents 110 are generally associated with the hosts 140, in this way, the agents 110 isolate the table 200 from general host 140 access. Preferably, each of the hosts 140 has a separate agent 110, so that each host has a separate mapping table 200. Alternatively, the system 100 could be configured so that more than one host 140 connects to an agent 110. If multiple hosts 140 connect to the same agent 110, the hosts 140 share access to the particular table 200.

The agent 110 stores the mapping table 200 in a volatile memory 111, typically DRAM. As a result, if one of the agents 110 shuts down or loses power, that agent 110 loses its copy of the table 200. For instance if the mapping agent 110 is embedded in the host system 140 and takes its power from that host system, as would be the case for a backplane card that serves as the mapping agent, the host 140 may cause the mapping agent 110 to shut down by eliminating power to the agent 110. However, by storing the table 200 in volatile memory, the table 200 can be easily and rapidly accessed and modified on the agents 110. Storing the mapping table 200 in volatile memory has the further advantage of

substantially reducing the cost and complexity of implementing the agents 110 as mapping components. Overall, the agents 110 allow the performance-sensitive mapping process to be parallelized and distributed optimally for performance.

The system 100 further comprises a controller 120 that, although separate from the agents 110, administers and distributes the mapping table 200 to the agents 110. Control of the mapping is centralized in the controller 120 for optimal cost, management, and other implementation practicalities. The controller 120 further stores the table 200 in a semi-permanent memory 121 so that the controller 120 retains the table 200 even after a power loss. The semi-permanent memory 121 is preferably a magnetic disk because of its high storage capacity and fast, frequently writing capabilities. The controller 120 may alternatively store the table 200 using other forms of programmable storage such as writeable optical media and electronically programmable memories. The controller 120 thus continues to store the table 200 even if the controller 120 shuts down or loses power.

In this way, the responsibility for persistent storage of the mapping tables 200 lies in the controller 120, consolidating both costs and complexity. The exact design of the controller 120 is not a subject of this disclosure. Instead, this disclosure focuses on the structure of the overall system and the interfaces between the mapping agent 110 and the controller 120. Accordingly, it should be appreciated that any controller, as known in the art of digital information storage, may be employed as needed to implement the present invention. Within this framework, each of the agents 110 preferably interacts with only the controller 120 and not with the other agents 110. As a result, the system 100 is highly scaleable and tolerant of component failures.

As described below, the interactions of the controller 120 and the agents 110 are defined in terms of functions and return values. In one embodiment of the virtual mapping system 100 illustrated in FIG. 3A, this

communication is implemented with messages on some sort of network transport, such as a communication channel 182. In another implementation of the system 100 depicted in FIG. 3H, the communication channel 182 is the storage network 130 itself. Any suitable technique may be used to translate commands, faults, and responses to network messages. The communication channel 130 may employ any type of known data transfer protocol such as TCP/IP. The particular interactions between the functions and activities of the controller 120 are described in greater detail below.

FIGS. 4A-4B schematically illustrate the contents of the table 200. As described above, the table 200 contains entries 210 (rows) that include a mapping between one or more virtual disk segments 220 and storage locations 230 on the storage containers 160. The storage locations 230 identify the particular storage container 160 and part of the storage container 160 that corresponds to the virtual disk index 220. The form for the storage locations 230 must be appropriate for the storage network being used. In a SCSI network, each of the storage locations 230 includes a LUN identifier 233 and a block identifier 235, also called an offset. All of the other fields in a mapping table entry 210 are simple integers or binary state values.

As depicted in FIG. 4A, the mapping table 200 may have one entry 210 per each "disk block" of the virtual disk 220. While possible to build, this would result in huge mapping tables and highly fragmented mapping, both of which introduce undesirable performance degradations. In another embodiment, each mapping table entry 210 represents a variable sized group of contiguous virtual disk blocks that map to contiguous blocks on one of the physical storage containers 160. This configuration of the table 200 offers greater mapping flexibility and dense mapping structures, but introduces algorithmic complexity in managing the variable sized extents and greater map entry lookup costs.

In response, the table 200 of a preferred embodiment uses mapping table entries 210, each having a fixed size number of contiguous blocks ("segments") on the virtual disk 150 that map to one storage container 160, as depicted in FIG. 4B. In this embodiment, each of the entries 210 contains a virtual disk segment 220 instead of a virtual disk block. The block identifier 235 likewise identifies a corresponding segment of actual storage blocks. While FIG. 4B illustrates the table 200 identifying an entire range of values for the mapping table entry 220 and the block identifier 235, the table 200 could likewise identify only the beginning or end block where the size of the actual and virtual storage segments is otherwise defined. While this configuration of FIG. 4B for the table 200 is possibly not as dense as variable sized extent mapping, the configuration offers the simplest and highest performance map access and space management. Regardless of the specifics of the table 200, the table 200 must map a virtual disk segment 220 to each physical storage block involved in I/O operations.

In a preferred embodiment, the system 100 has multiple tables 200, each having different mappings between a virtual disk 150 and the storage containers 160. In this way, different hosts 140 may have different access to the same storage container 160. Where the table 200 does not include one of the storage locations 230, hosts 140 using this table (i.e., the hosts 140 connecting to the agent 110 that stores this table) cannot access information stored at the storage location. In fact, the host 140 will not realize that this storage location 230 exists.

During operation, the host 140 issues an I/O operation (e.g., read or write) to some block or blocks on a virtual disk 150. Each virtual memory block is represented in the mapping table 200, either as an individual entry or as part of a virtual disk segment 220. Each block contained in the I/O operation is mapped to the appropriate location on the storage container 160. The mapping agent 110 issues a corresponding I/O operation issued to the storage container 160. The I/O operation results
14

are then collected and presented as a completed operation on the virtual disk 150.

In addition to mapping information specifying the storage location, each mapping table entry 210 also contains several states. The states are Boolean variables that provide information on the current status of the virtual disk segment. These states are important because they allow the table 200 stored in the agent 110 to be remotely loaded and manipulated from the controller 120. These states and interfaces provide the ability for the mapping tables to be distributed and for the mapping table entries to be volatile. The disclosure first describes the states and, then, explains some of the functions for the states. The table 200 includes at least an invalid state 240 indicating whether any I/O operations may occur on the virtual disk segment 220 and its corresponding physical location 230. The invalid state may be activated during a first I/O operation to prevent further I/O operations until completion of the first I/O operation. In a preferred embodiment, the table 200 further includes a no-write (Nw) state 250 that indicates whether the data contained at the corresponding physical location 230 may currently be changed. The Nw state 250 allows for improved storage system performance because it permits data to be read during another I/O operation. The invalid state 240 and the Nw state 250 function during dynamic loading of mapping table entries, dynamic mapping changes, volatility of mapping table entries, and data sharing among similar virtual disks 150.

When activated, the invalid state 240 generally indicates that the mapping table entry 210 contains no useable mapping information and cannot support I/O operations. Any attempt to implement an I/O operation through this table entry 210 causes the mapping agent 110 to send a fault message to the controller 120. The agent 110 does not proceed with the I/O operation until the controller 120 returns a fault response. In one embodiment, the system 100 initially activates the invalid state 240 for all entries 210 in the table 200 when the table 200 is newly created. In this way, the table 200 ignores any residual entries in memory from prior

stored tables to insure that current entries are active and reliable. Similarly, the invalid state 240 may be activated when entry 210 is "forgotten" and lost by the mapping agent 110 volatile memory. If the invalid state 240 is activated in the entry 210, then all other values and states in the entry 210 are assumed to contain no valid information and are ignored.

Because the tables 200 located in the mapping agents 110 are volatile, any failure or restart of the mapping agents 110 causes all of the entries 210 to have an active invalid state 240. A sustained loss of communication between the controller 120 and mapping agent 110 also causes I/O operations to stop, either by making all mapping table entries revert to an active invalid state 240 or by adding additional mechanisms to suspend I/O operations until directed by the controller 120 to resume I/O operations. This configuration allows the controller 120 to continue coordinating other mapping agents 110 by knowing that a failed or unreachable mapping agent 110 has been placed into a known state, providing the controller 120 high availability of data access to the surviving mapping agents 110.

As presented above, the Nw state 250, when activated, indicates that any write operations to the virtual disk segment(s) 220 represented by the entry 210 cause the agent 110 to send a fault message the controller 120. The agent 110 does not allow the host 140 to write to the storage locations 230 until the controller 120 returns a fault response to deactivate the Nw state 250 or until the system 100 otherwise takes some action to write to a segment despite the active Nw state 250. Unlike the invalid state 240, the activated Nw state 250 prevents operations from generating faults. Instead, the agent 110 generally allows the host 140 to proceed to access data at the storage location 230. Accordingly, if only the Nw state is activated, table entry 210 must contain a useable storage location 230.

In another embodiment, the table 200 further includes a zero (Z) state 260. When active, the Z state 260 indicates that the virtual disk segment 220 represented by the entry 210 contains all zero bytes. This feature allows a virtual disk 150 to be created and appear to be initialized without the need to allocate or adjust any underlying non-virtual storage. If an entry 210 contains an active Z state 260, the agent 110 ignores the storage address 230. If the host 140 attempts to read information stored at storage address 230, the agent 110 returns only zero-filled blocks regardless of the actual contents of the storage address 230. On the other hand, any attempts to write data at the storage address 230 when Z state 260 is active cause the agent 110 to send a fault message to the controller 120. The agent 110 does not allow the host 140 to write to the storage locations 230 until the controller 120 returns a fault response that deactivates the Z state 260 or until the system 100 otherwise takes some action to write to a segment despite the active Z state 260.

In another configuration, the mapping table 200 further includes an error (E) state 270. When active, the E state 270 indicates the existence of an error condition and provides the information necessary to instruct the agent to return an error without disrupting any previous state. The E state 270 is used where a pre-existing failure is known and such failure would cause any attempts at I/O access to fail. It should be noted, however, that the E state 270 could also be used as the means to issue an error status from a mapping fault. If an entry 210 contains an active E state 270, the agent 110 ignores the storage address 230. If the host 140 attempts to read from or write to the storage address 230, the agent 110 returns an error to the host 140.

The interaction of the agent 110 and the controller 120 is now described in greater detail. In one category of interactions, fault/response operations, the agent 110 sends a message to the controller 120 indicating the occurrence of a fault during an I/O operation to the table 200. Typically, the fault occurs as a result of an activated state (as described above) that prevents the execution of the I/O operation by the agent. The

agent 110 sends the fault message to the controller 120. The controller then determines an appropriate action and commands the agent 110 accordingly.

In one type of fault/response operation, a map fault, the mapping agent 110 alerts the controller 120 that an I/O operation requested by the host 140 cannot be completed because the mapping table entry 210 has an activated state that prevents the completion of the requested I/O operation. For example, the mapping agent 110 produces a fault message to the controller 120 in response to a request for any I/O operation to a table entry 210 having an active invalid flag 240 or an attempt to write to storage address 230 having an active corresponding Nw flag 250. The map fault message from the agent 110 generally identifies the requested I/O operation, the virtual disk segment 220 involved, and the table state preventing the I/O operation. When the fault occurs, the agent does not attempt to carry out the I/O operation. Instead, the controller 120 uses the fault message to respond to the faulted I/O operation (e.g. load map entry, change map entry, delay until some other operation has completed). The controller 120 response informs the mapping agent 110 how to proceed to overcome the cause for the fault.

The controller 120 generally instructs the agent 110 either to resolve the problem or to send an error message to the requesting host. When resolving the problem, the controller 120 sends a replacement table entry 210. The agent 110 then inserts the new table entry 210 in the table (in place of the former faulty entry) and then retries I/O operation. If the controller 120 cannot resolve the problem, it instructs the agent 110 to issue an error message to the host. To cause the agent 110 to issue an error message, the controller instructs the agent to activate the error state 260 for the table entry 210 causing the fault. As described above, the agent 110 then issues an error message to the host 140 regardless of the other contents of the table entry 210.

Commands to the agent 110 initiated by the controller 120 comprise a second category of interactions, command/response operations. Among these commands initiated by the controller 120 include the creation of a new mapping table 200 with all entries set to have an active invalid flag or the deletion of an existing table 200. The controller 120 may obtain from the agent 110 the contents of one of the entries 210 or the status of the one of the states in this entry 210. The controller 120 can further order the agent 110 to set all of the contents for one of the entries 210 or the status of one of the states for the entry 210. It should be noted that once the invalid state 240, the error state 260, and the zero state 270 are active, the controller 120 cannot deactivate the state because, as described above, initial activation of these states voids the storage address 230. To deactivate these states, the controller 120 must instruct the agent 110 to replace the existing entry 210 with an entirely new entry. To each of these commands, the agent 110 returns a response to the controller 120 after completing the ordered task.

When the controller 120 instructs the agent to either set or obtain information from the table 200, the system optimally allows the controller 120 to specify multiple, contiguous map table entries 210 in a single command. This arrangement allows the agent 110 and the controller 120 to interact more efficiently and with fewer instructions.

When the controller 120 commands the agent 110 to set one or all of the values and states in the table entry 210, the controller 120 command to the agent 110 optimally includes a blocking flag. The blocking flag instructs the agent 110 to delay responding to the controller 120 command until after the completion of any I/O operations initiated before the command. The blocking flag applies to only that command in which it resides. Other concurrent commands and subsequent commands are not generally affected by the blocking flag of one command. In particular, the agent 110 immediately changes the table 200, as instructed in the command, but does not notify the controller 120 of this change until completing all previously existing I/O operations. In this way, the agent

110 signals to the controller 120 the completion of all I/O operations using the former table 200 that do not reflect the changes to the table specified in the command.

During a majority of the operation, the mapping agent 110 operates without faults. In non-fault cases, i.e. the mapping table entries 210 are valid and do not have any active states that prevent the requested I/O operation, the virtual disk I/O operates entirely through the mapping agent 110. Thus, all I/O operations proceed through the mapping table 200 and directly to the physical storage containers 160 without any involvement of the controller 120. As a result, the controller 120 inserts itself into an I/O stream only when needed to perform various management operations and typically does not become involved in non-faulting cases, allowing the system 100 to have high performance and scalability.

The foregoing description of the preferred embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto. The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.

4. Brief Description of Drawings

FIGS. 1A-1B ~~PRIOR ART~~ are known systems for connecting a host to a storage device;

FIGS. 2A-2C ~~PRIOR ART~~ are known virtualized storage area networks;

FIGS. 3A-3B ³⁻⁴ are schematic illustrations of a distributed virtual storage area network in accordance with embodiments of the present invention; and

FIGS. 4A-4B ⁵⁻⁶ are schematic illustrations of a mapping table for use in the distributed virtual storage area network of FIG. 3 in accordance with an embodiment of the present invention.

2 /

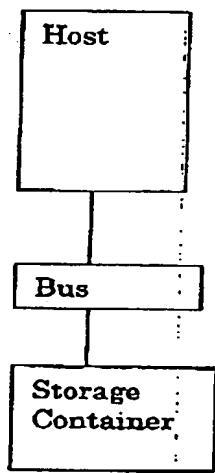


FIG. 1A

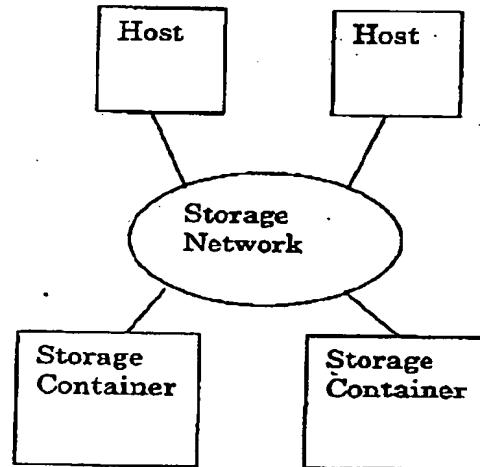


FIG. 1B

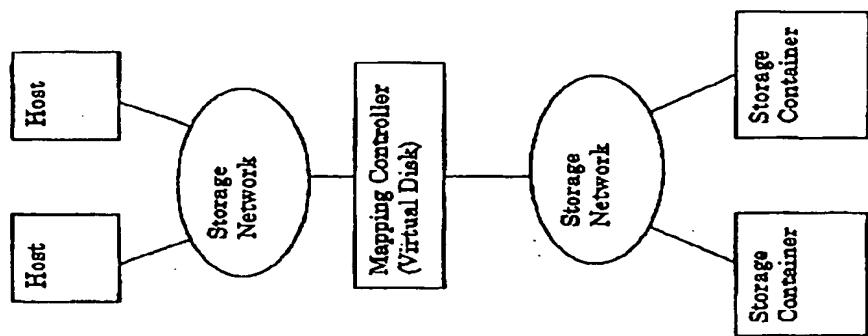


FIG. 2C

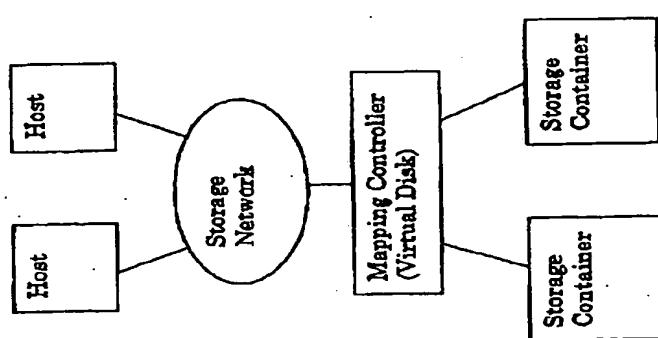


FIG. 2B

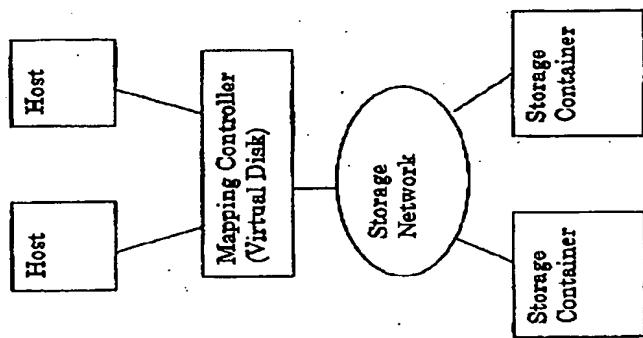


FIG. 2A

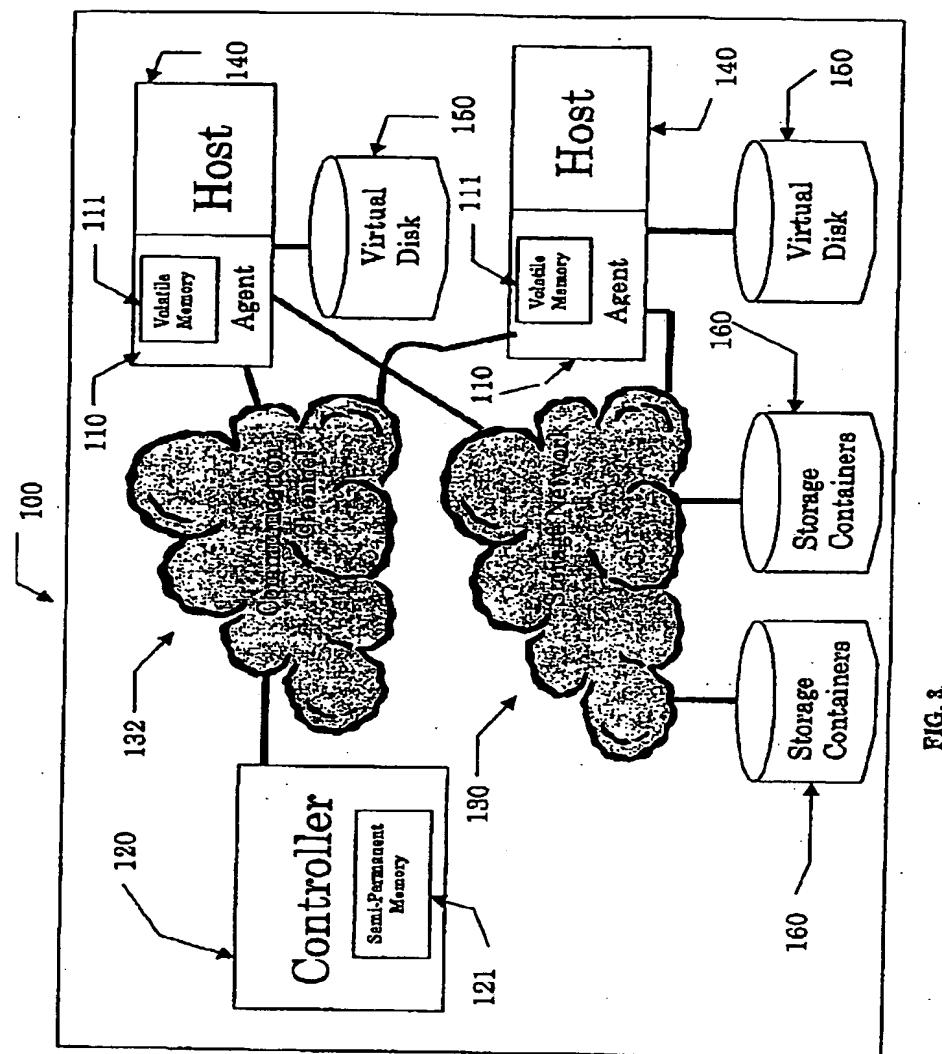


FIG. 3

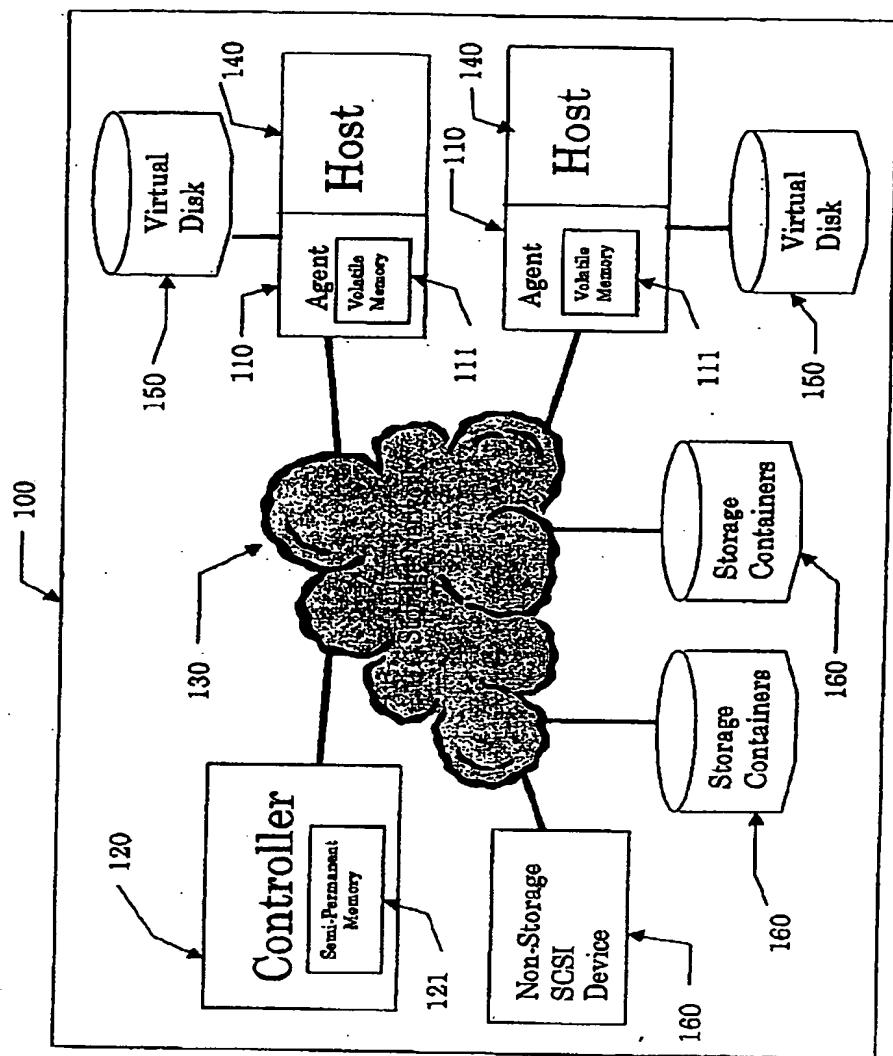


FIG. 4.

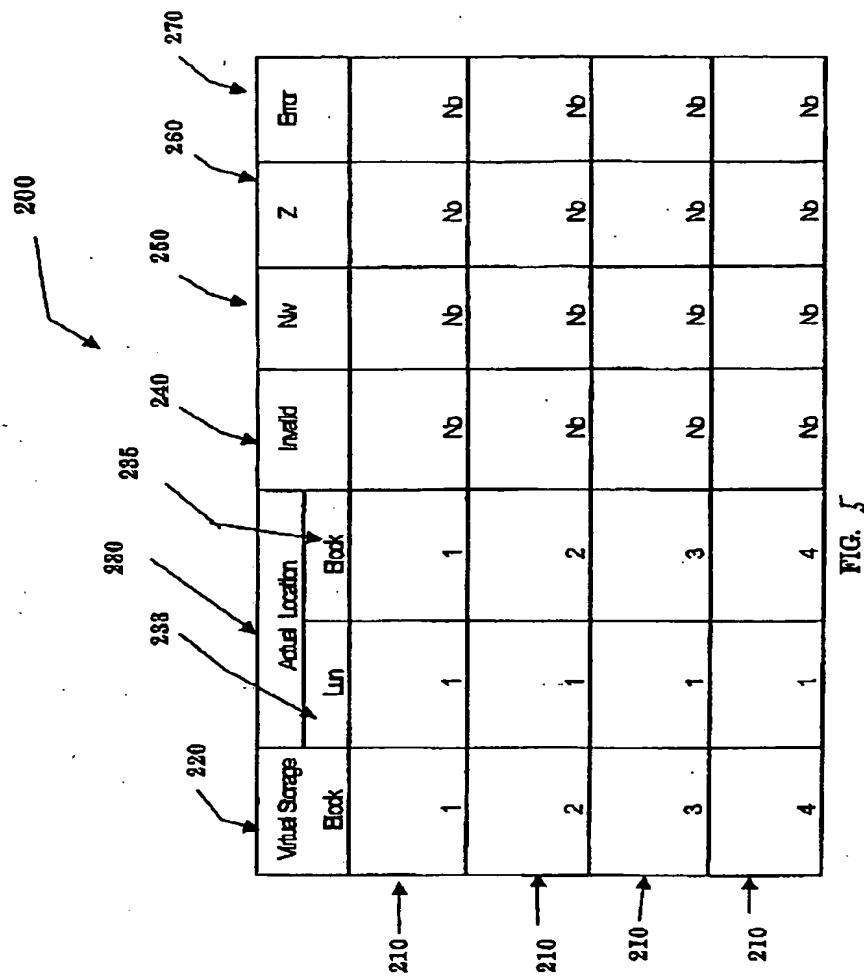


FIG. 5

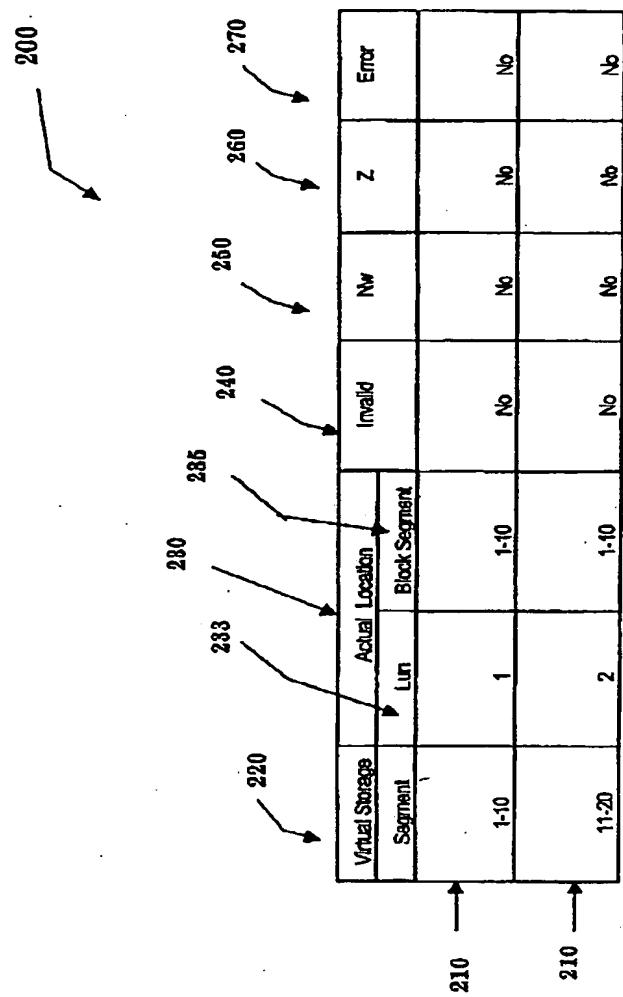


FIG. 6

1. Abstract

The present invention provides a system and method for creating virtualized storage in a storage area network using distributed table-driven input/output mapping. The present invention distributes the virtualization mapping in multiple parallel, mapping agents that are separate from a controller. This allows the performance-sensitive mapping process to be parallelized and distributed optimally for performance, while the control of the mapping may be located in a controller chosen for optimal cost, management, and other implementation practicalities. The mapping agents store the virtual mapping tables in volatile memory, substantially reducing the cost and complexity of implementing the mapping agents. The controller is responsible for persistent storage of mapping tables, thereby consolidating the costs and management for persistent mapping table storage in a single component. Distributed virtualization also allows the controller to manage multiple virtual disks used by multiple host systems, and allows a single virtual disk to be shared by multiple host systems. The mapping agents preferably do not interact only with other mapping agents, thereby improving the scalability of the virtual storage system and the virtual storage system's tolerance of component failures.

2. Representative Drawing

Fig. 3